

OpenMHD: Godunov-type code for magnetic reconnection and MHD problems

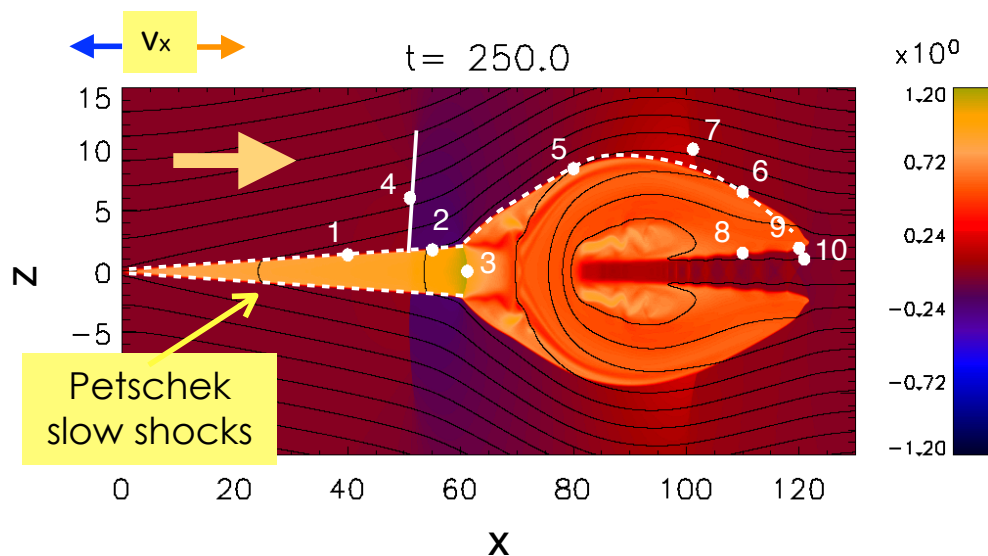
Seiji ZENITANI

Kobe University

[1] Zenitani, *Phys. Plasmas* **22**, 032114 (2015)

[2] Zenitani, *Astrophysics Source Code Library*, ascl:1604.001 (2016)

MHD shocks in reconnection



2016

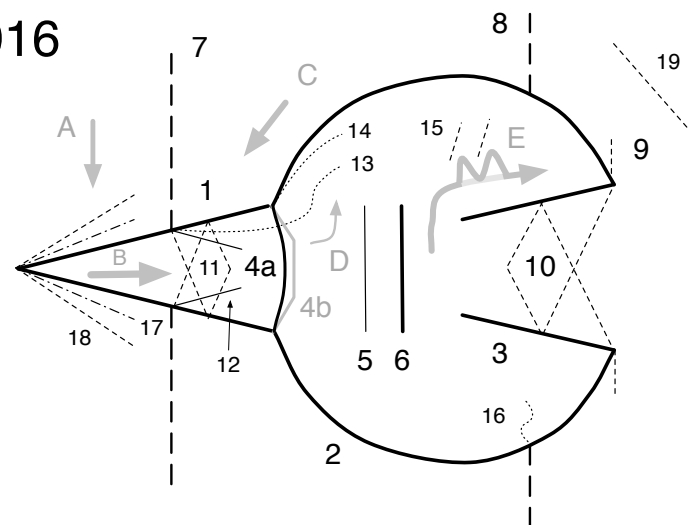


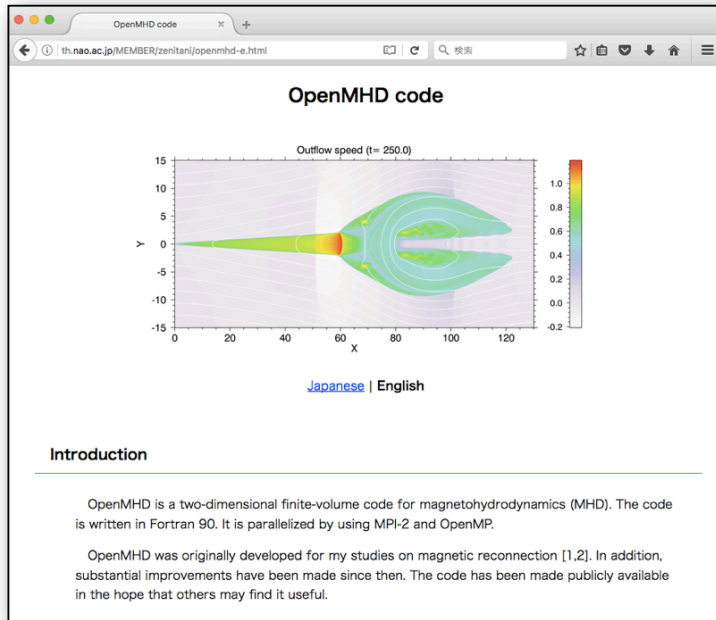
TABLE I. Rankine-Hugoniot analysis. The subscripts 1 and 2 denote the upstream and downstream quantities. The locations (x,z) in the simulation domain [see also Fig. 1(b)], the shock normal vector \hat{n} , the shock velocity v_{sh} , the angle between \hat{n} and the upstream magnetic field B_1 , the upstream plasma beta, flow Mach numbers to fast, intermediate (Alfvén), and slow-mode speeds, and the temperature ratio. The asterisk sign (*) indicates unreliable results (see Sec. III F). The letter (S) indicates a slow shock, (F) is a fast shock, and (U) is unclassified.

No.	Location	(n_x, n_z)	v_{sh}	$ \theta_{B1} $	β_1	M_{f1}	M_{i1}	M_{s1}	M_{f2}	M_{i2}	M_{s2}	T_2/T_1	
1	(40.0, 1.35)	(-0.03, 1.00)	0.0	86.3	0.22	0.06	0.98	2.49	0.04	0.69	0.69	2.72	(S) Petschek shock
2	(55.0, 1.75)	(-0.04, 1.00)	-0.013	86.3	0.098	0.06	0.88	3.22	0.04	0.58	0.58	4.58	(S) Petschek shock
3	(61.2, 0)	(-1.00, 0.00)	-0.40	90	303	1.41			0.77			1.38	(F) Reverse shock
4	(51.0, 6.0)	(1.00, -0.04)	0.31	9.4	0.12	0.41	0.42	1.34	0.33	0.34	0.78	1.33	(S) Postplasmoid vertical shock
5	(80.0, 8.4)	(-0.18, 0.98)	-0.06	86.5	0.16	0.05	0.85	2.47	0.03	0.56	0.65	2.54	(S) Outer shell
6	(110.0, 6.5)	(0.24, 0.97)	0.19	84.9	0.21	0.06	0.76	1.99	0.05	0.53	0.64	2.06	(S) Outer shell
7	(101.2, 10.0)	(0.94, 0.33)	0.54	25.2	0.23	0.43	0.49	1.15	0.39	0.44	0.87	1.15	(S) Forward vertical shock
8	(110.0, 1.5)	(-0.06, -1.00)	0.10	87.8	1.1	0.12	4.5*	6.5*	0.12	3.9*	4.0*	1.55	(U) Intermediate shock?
9	(120.0, 1.9)	(0.13, -0.99)	0.13	87.1	0.49	0.09	2.0*	3.8*	0.08	1.7*	1.9*	1.86	(U) Slow shock?
10	(120.9, 1.0)	(0.64, -0.77)	0.50	46.8	2.63	1.22	3.00	3.40	0.88	2.66	3.06	1.18	(F) Oblique shock

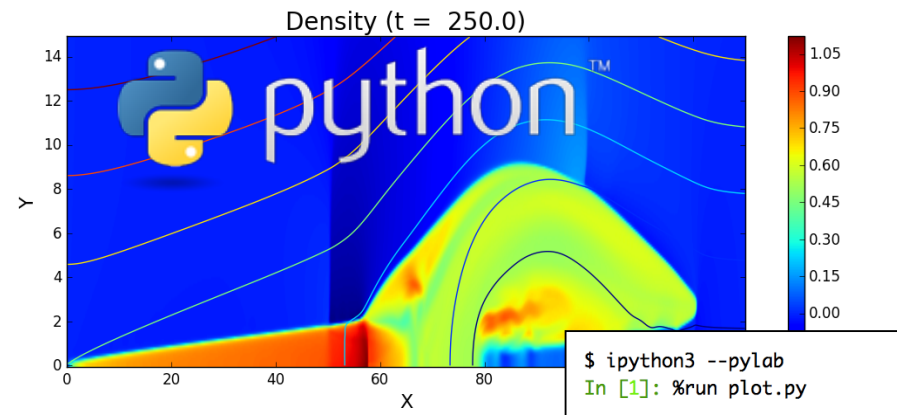
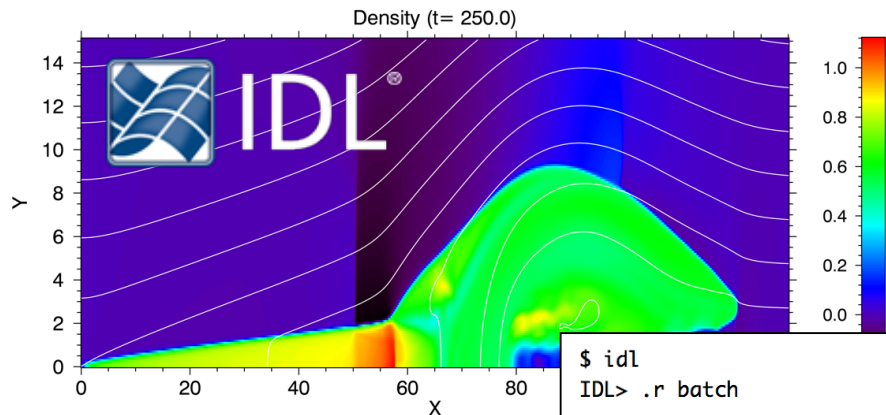
1. Petschek slow shock (Petschek 1964)
 2. outer shell = slow shock (Ugai 1995)
 3. intermediate shock (Abe & Hoshino 2001) or slow shock (Saito et al. 1995)
 - 4a fast shock (Forbes & Priest 1983)
 - 4b oblique shock & Mach disk (Takasao et al. 2015)
 5. looptop front (Ugai 1987)
 6. tangential discontinuity
 7. post-plasmoid vertical slow shock (Zenitani et al. 2010)
 8. outer vertical slow shock (Zenitani & Miyoshi 2011)
 9. fast-mode wave front (Saito et al. 1995)
 10. overexpanded shock-diamonds (Zenitani et al. 2010)
 11. underexpanded shock-diamonds (Zenitani 2015)
 12. slow expansion wavefront (Zenitani 2015)
 13. contact discontinuity (Zenitani & Miyoshi 2011, 2015)
 14. contact discontinuity (Zenitani 2015)
 15. vortex-driven shocklets (Miura 1992, 1995)
 16. contact discontinuity (Zenitani 2015)
 17. rotational discontinuity [in guide-field reconnection] (Petschek & Thorne 1967)
 18. conduction front [with heat conduction] (Yokoyama & Shibata 1997)
 19. forward head shock [in asymmetric reconnection] (Nitta et al. 2016)
- A. reconnection inflow
 B. outflow jet
 C. post-plasmoid reverse flow
 D. internal flow
 E. flapping jet (KH instability)

Zenitani & Miyoshi 2011
 Zenitani 2015

OpenMHD - public code



- MHD code by Zenitani & Miyoshi (2011, 2015) with recent improvements
- Fortran 90, 2D (Internally 3D), HLLD, MUSCL, MPI+OpenMP, MPI-IO, GPU (CUDA)
- Visualization code in IDL and Python3 (matplotlib)



<https://sci.nao.ac.jp/MEMBER/zenitani/openmhd-e.html>

MHD equations

- Resistive MHD equations

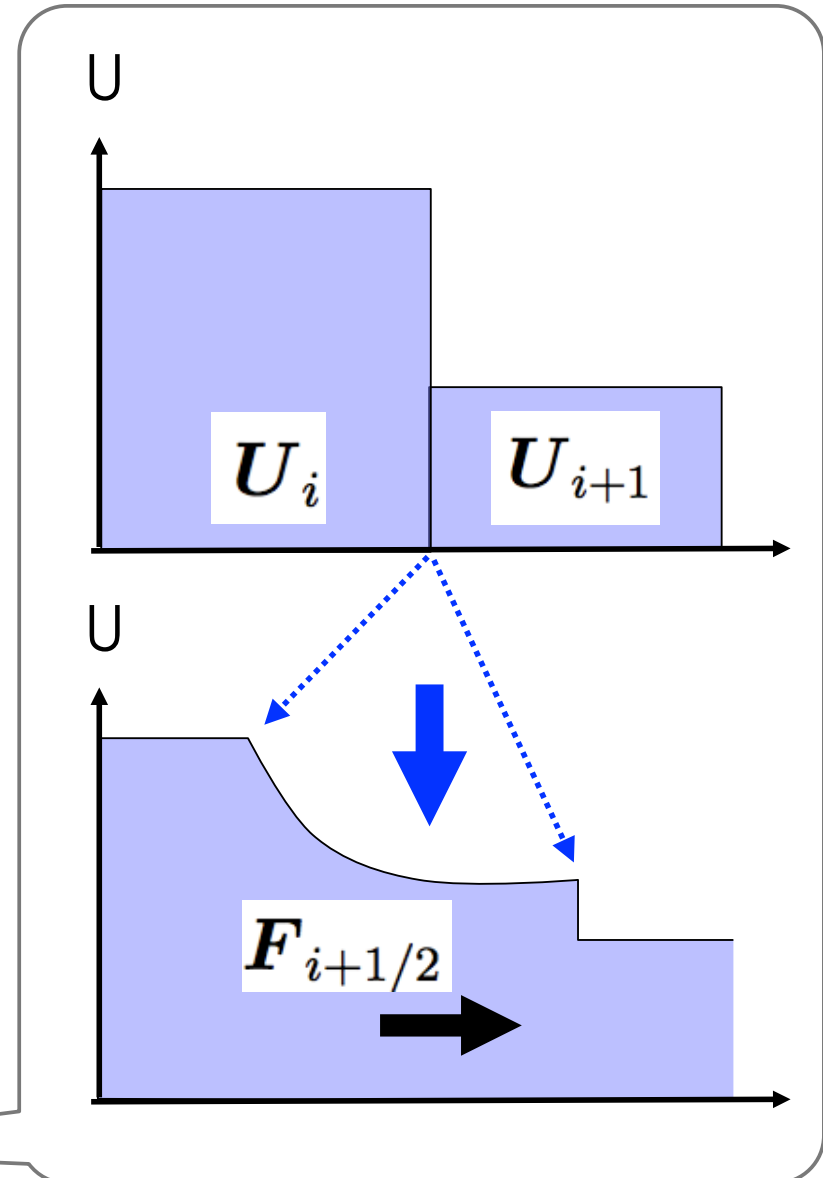
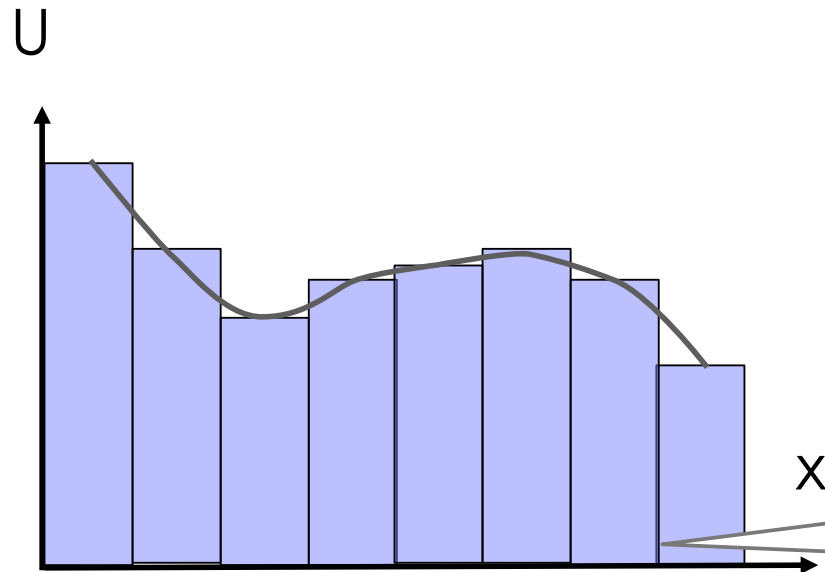
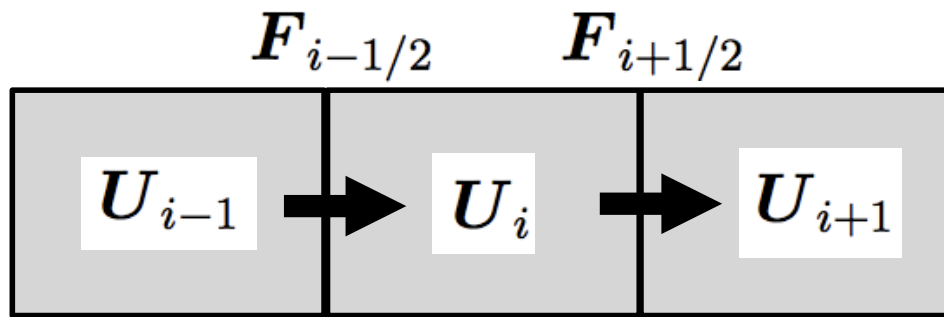
$$\begin{aligned}\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) &= 0, \\ \frac{\partial \rho \vec{v}}{\partial t} + \nabla \cdot (\rho \vec{v} \vec{v} + p_T \overleftrightarrow{I} - \vec{B} \vec{B}) &= 0, \\ \frac{\partial e}{\partial t} + \nabla \cdot \left((e + p_T) \vec{v} - (\vec{v} \cdot \vec{B}) \vec{B} + \eta \vec{j} \times \vec{B} \right) &= 0, \\ \frac{\partial \vec{B}}{\partial t} + \nabla \cdot (\vec{v} \vec{B} - \vec{B} \vec{v}) + \nabla \times (\eta \vec{j}) + \nabla \psi &= 0, \\ \frac{\partial \psi}{\partial t} + c_h^2 \nabla \cdot \vec{B} &= - \left(\frac{c_h^2}{c_p^2} \right) \psi,\end{aligned}$$

- Ohm's law

$$\mathbf{E} + \mathbf{v} \times \mathbf{B} = \eta \mathbf{j}$$

- Hyperbolic divergence cleaning (Dedner+ 2002)
- Second-order TVD RK + Strang splitting

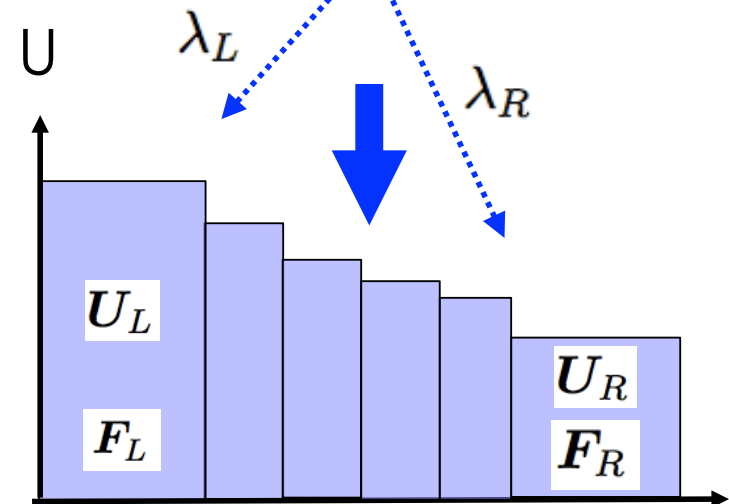
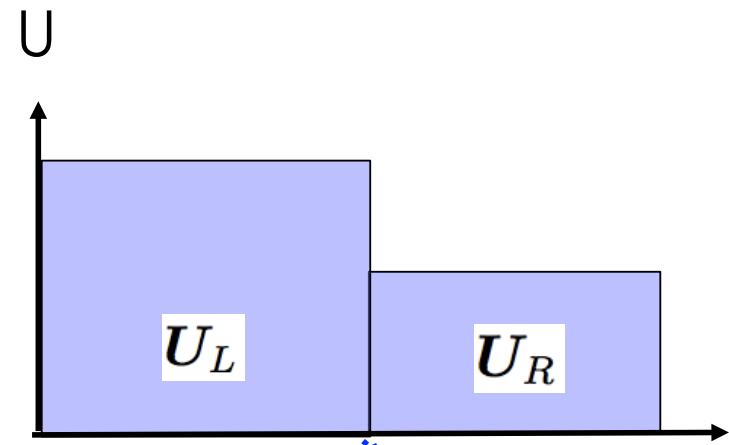
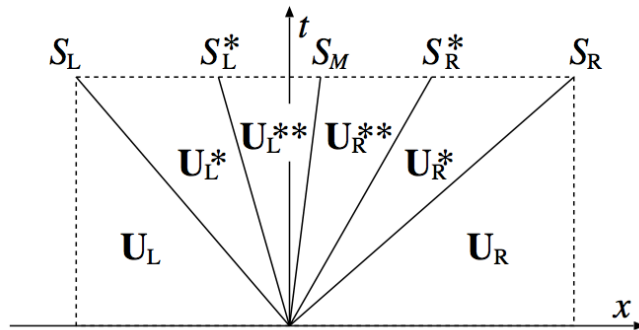
Riemann solver



HLLD solver

- Proposed by Miyoshi & Kusano 2005
- An advanced variant of HLL solver

$$\mathbf{F}_{\text{HLLD}} = \begin{cases} \mathbf{F}_L & \text{if } S_L > 0, \\ \mathbf{F}_L^* & \text{if } S_L \leq 0 \leq S_L^*, \\ \mathbf{F}_L^{**} & \text{if } S_L^* \leq 0 \leq S_M, \\ \mathbf{F}_R^{**} & \text{if } S_M \leq 0 \leq S_R^*, \\ \mathbf{F}_R^* & \text{if } S_R^* \leq 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R < 0. \end{cases}$$

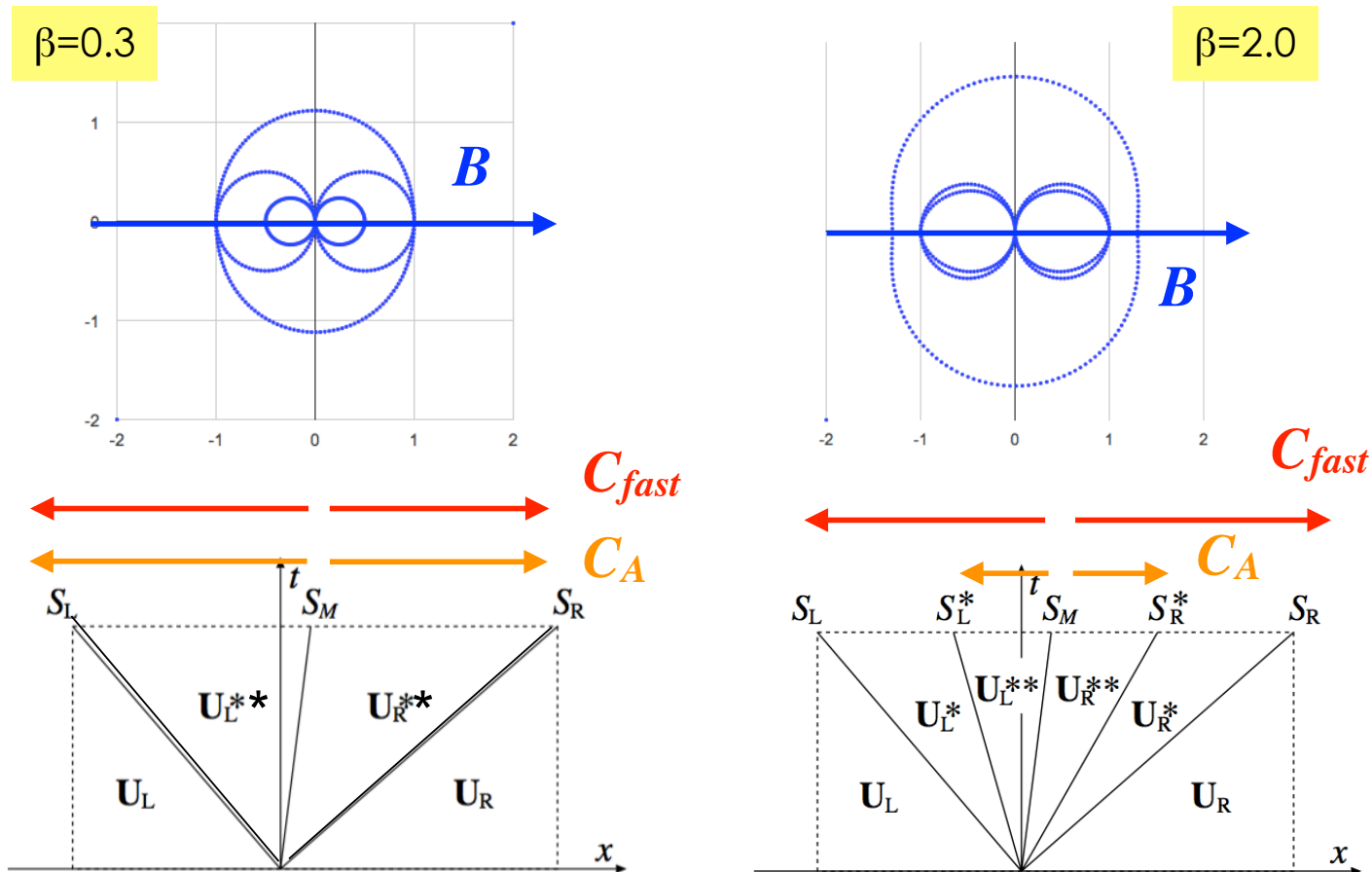


A problem in the HLLD scheme

- In the parallel direction, $C_A = C_{fast}$ for $\beta < 1.2$.
- RD sometimes outruns the Riemann fan.
- We offer the problem parameters.

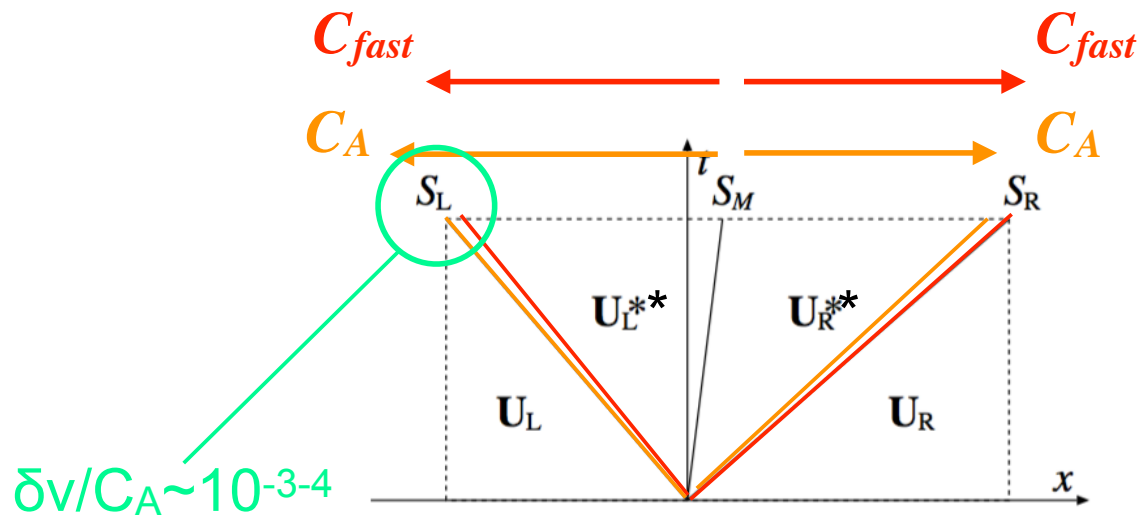
```

! left
vx0 = -3.550886791794632E-002
vy0 = 0.202198834859736
vz0 = 0.d0
pr0 = 0.573857561003025
ro0 = 1.46274860814133
bx0 = 1.d0
by0 = 0.535958263440366E-004
bz0 = 0.d0
! right
vx0 = -3.563873070012032E-002
vy0 = 0.201451798427819
vz0 = 0.d0
pr0 = 0.575034711174729
ro0 = 2.46444099174925
bx0 = 1.d0
by0 = -4.588121811862009E-004
bz0 = 0.d0
    
```



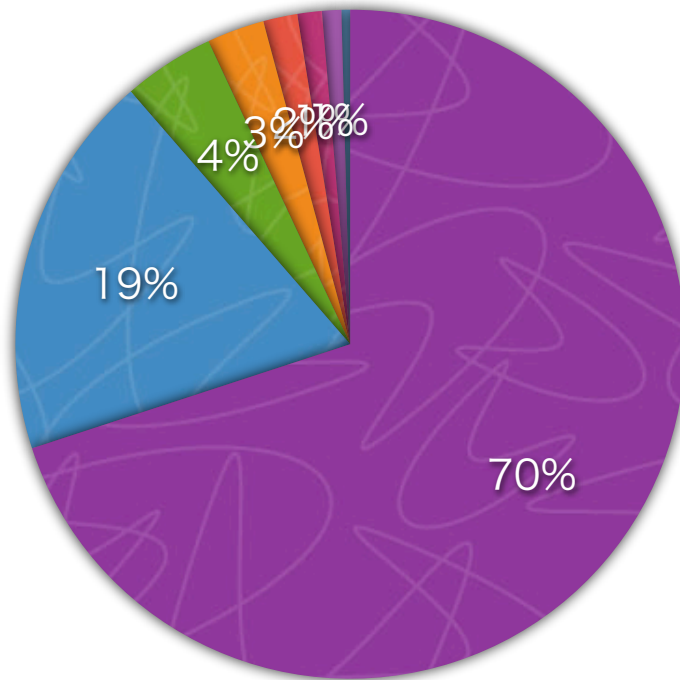
Solutions

- 1. HLLC=HLLD hybrid solver
 - Positivity
 - If clauses - not ideal for optimization
- 2. Faster C_{fast} (Miyoshi 2015 private comm.)
- 3. Adjust wavespeeds: $S_L^* = \max(S_L, S_L^*)$, $S_R^* = \min(S_R^*, S_R)$
 - Good for optimization
 - Theoreticians are unhappy

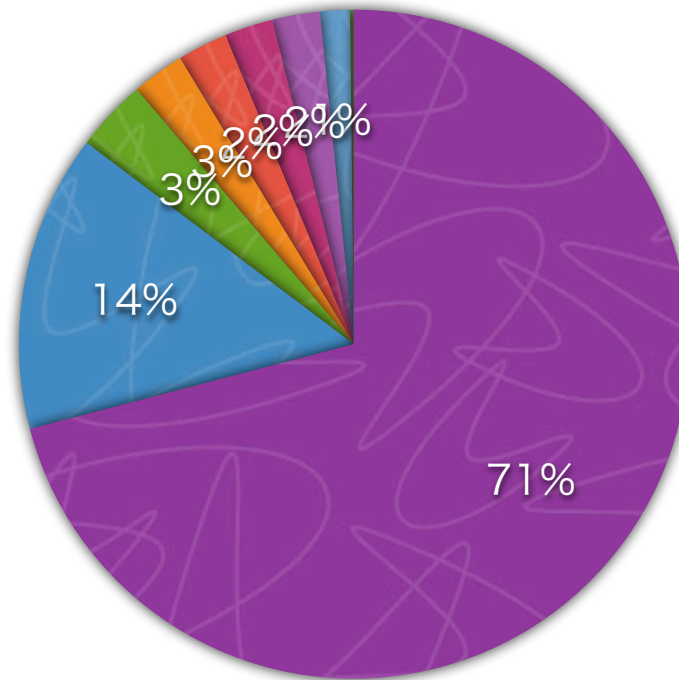


OpenMHD/2019 (HLLD) - Profiler output

Orszag-Tang vortex



Magnetic reconnection



- flux_solver
- limiter
- set_dt
- u2v
- tvd_rk22
- tvd_rk21
- flux_glm
- other

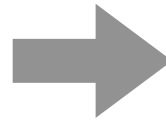
- flux_solver
- limiter
- set_dt
- tvd_rk22
- flux_resistive
- tvd_rk21
- u2v
- flux_glm
- other

gfortran 4.8

Performance tuning (1/2)

- Loop indexes
- Specifying `intent` attributes in subroutines
 - In particular, `intent(in)` and `intent(out)` work great.

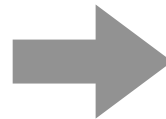
```
subroutine bc(U,ix,jx)
  implicit none
  include 'param.h'
  integer :: ix, jx
```



```
subroutine bc(U,ix,jx)
  implicit none
  include 'param.h'
  integer, intent(in) :: ix, jx
```

- Reducing `if`-statements
 - This does not improve the performance on Intel computers, but it works well on FX computers
 - minmod limiter:

```
if( gA*gB .le. 0 ) then
  grad = 0.d0
else
  if( gA .gt. 0 ) then
    grad = 0.5d0 * min(gA,gB)
  else
    grad = 0.5d0 * max(gA,gB)
  endif
endif
```



```
grad = (sign(0.25d0,gA)
+sign(0.25d0,gB))*min(abs(gA),abs(gB))
```

- MC limiter:

```
grad = (sign(0.5d0,gA)+sign(0.5d0,gB))*min(abs(gA),abs(gB),0.25d0*abs(gA+gB))
```

Performance tuning (2/2)

- Reducing square roots and divisions

- [before] Fastest fast-mode speed

```
vfL = sqrt( ( (f1+B2) + sqrt(max( (f1+B2)**2-f2, 0.d0 )) ) / ( 2*VL(i,j,ro) ) )
vfR = sqrt( ( (f1+B2) + sqrt(max( (f1+B2)**2-f2, 0.d0 )) ) / ( 2*VR(i,j,ro) ) )
f1 = max( vfL, vfR )
```

- [after]

```
aL = ( (f1+B2) + sqrt(max( (f1+B2)**2-f2, 0.d0 )) ) / ( 2*VL(i,j,ro) )
aR = ( (f1+B2) + sqrt(max( (f1+B2)**2-f2, 0.d0 )) ) / ( 2*VR(i,j,ro) )
f1 = sqrt( max( aL, aR ) )
```

- Using register variables
- Using collapse option in OpenMP directives

```
!$omp parallel do private(i,j,k) collapse(2)
  do k=1,var1
    do j=2,jx-1
      do i=2,ix-1
```

- Using good compilers

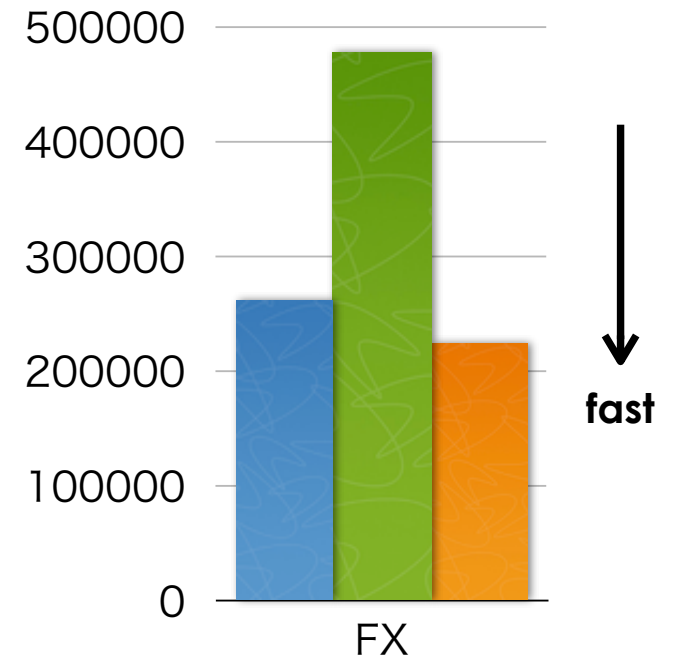
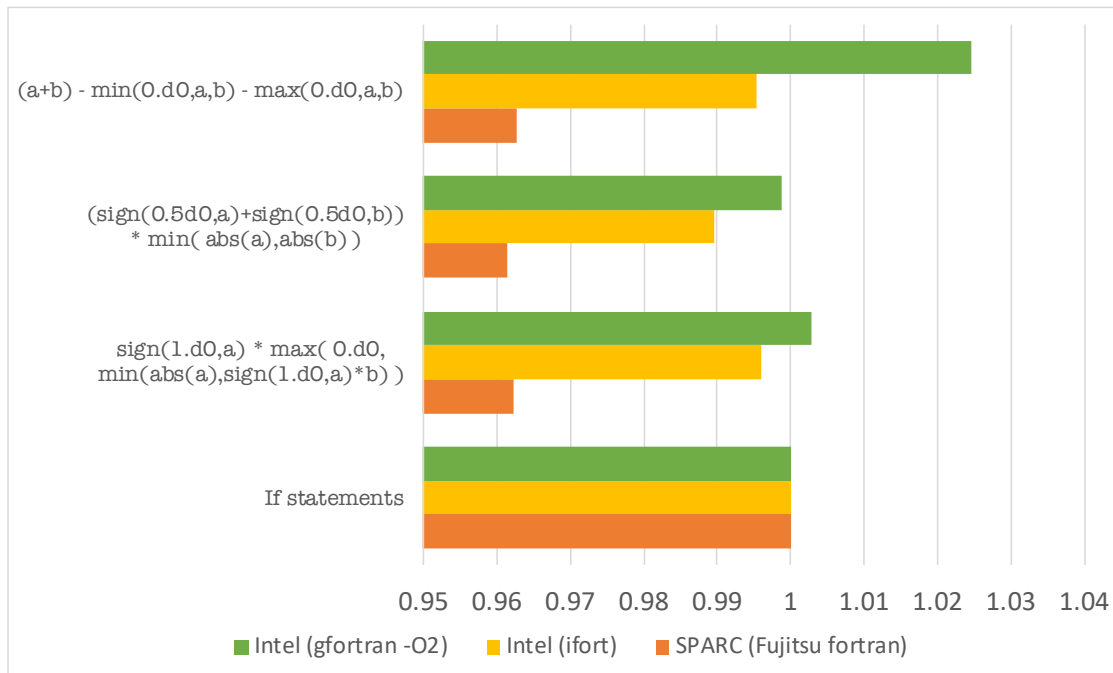
- Intel Fortran compiler is usually better than gfortran
- Gfortran 5.4 or 6.x is substantially better than gfortran 4.8

Improving performance - slope limiters

minmod

MC

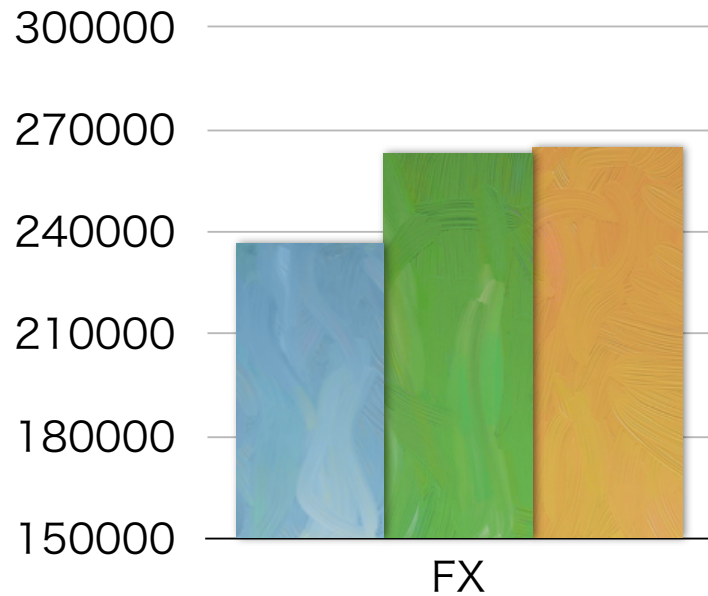
fast ←



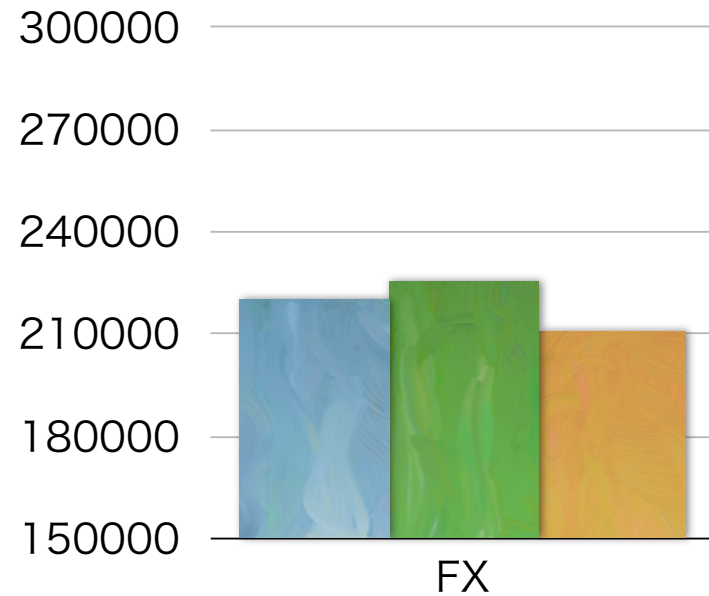
- If statements
- $\text{minmod}(2a, \text{minmod}(2b, c))$
- $(\text{sign}(0.5d0,a) + \text{sign}(0.5d0,b)) * \min(\text{abs}(2a), \text{abs}(2b), \text{abs}(c))$

Improving performance - slope limiters

If statements



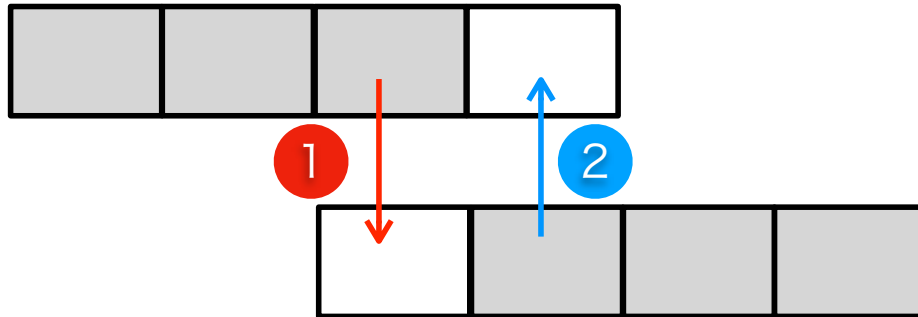
min, sign etc.



↓
fast

■ minmod ■ MC ■ Koren

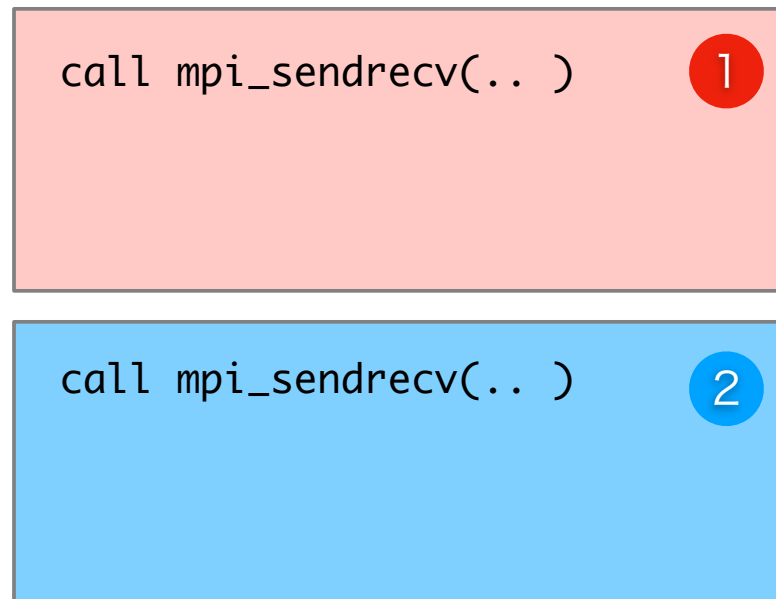
MPI nonblocking communication



- Other operation while sending/receiving data (communication hiding)

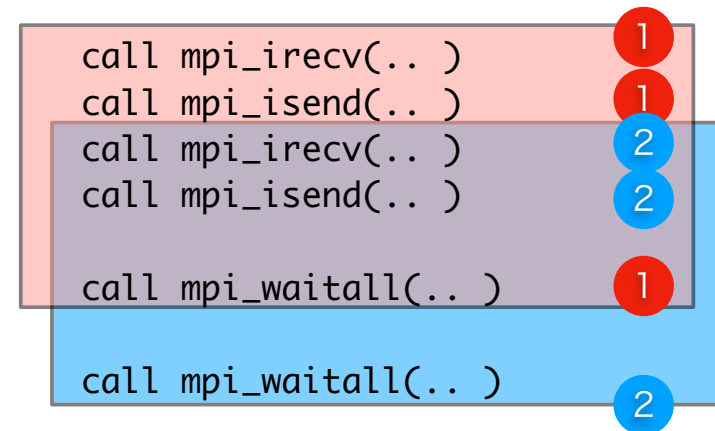
time

(blocking)



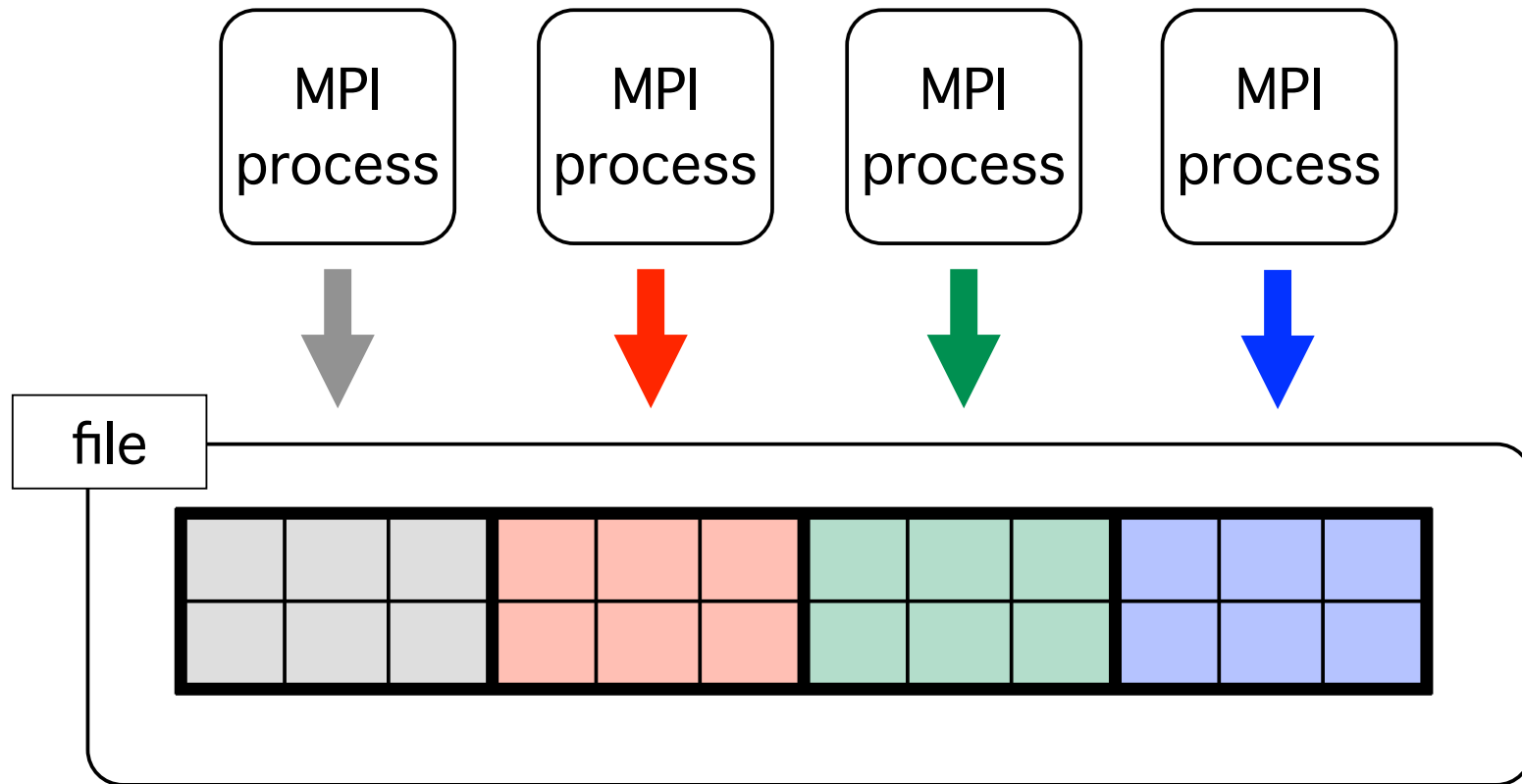
(nonblocking)

time



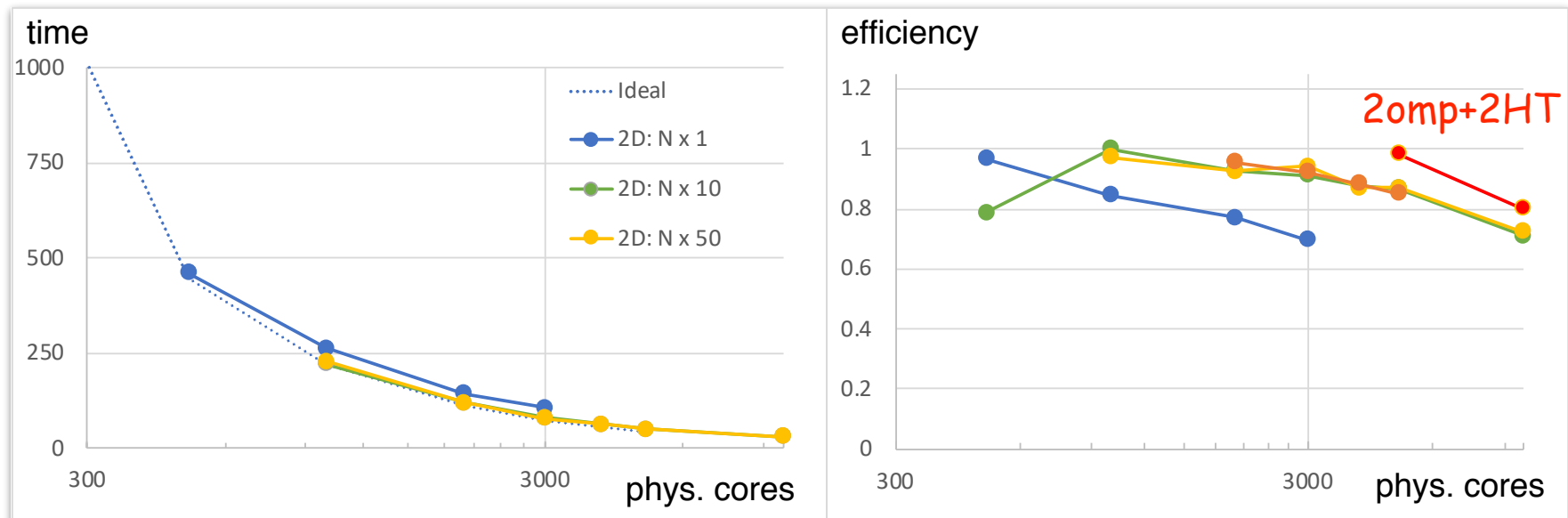
Parallel I/O (MPI-IO)

- Multiple MPI processes concurrently access data in a single file.
- The way to access the shared file is defined by `MPI_File_set_view`.

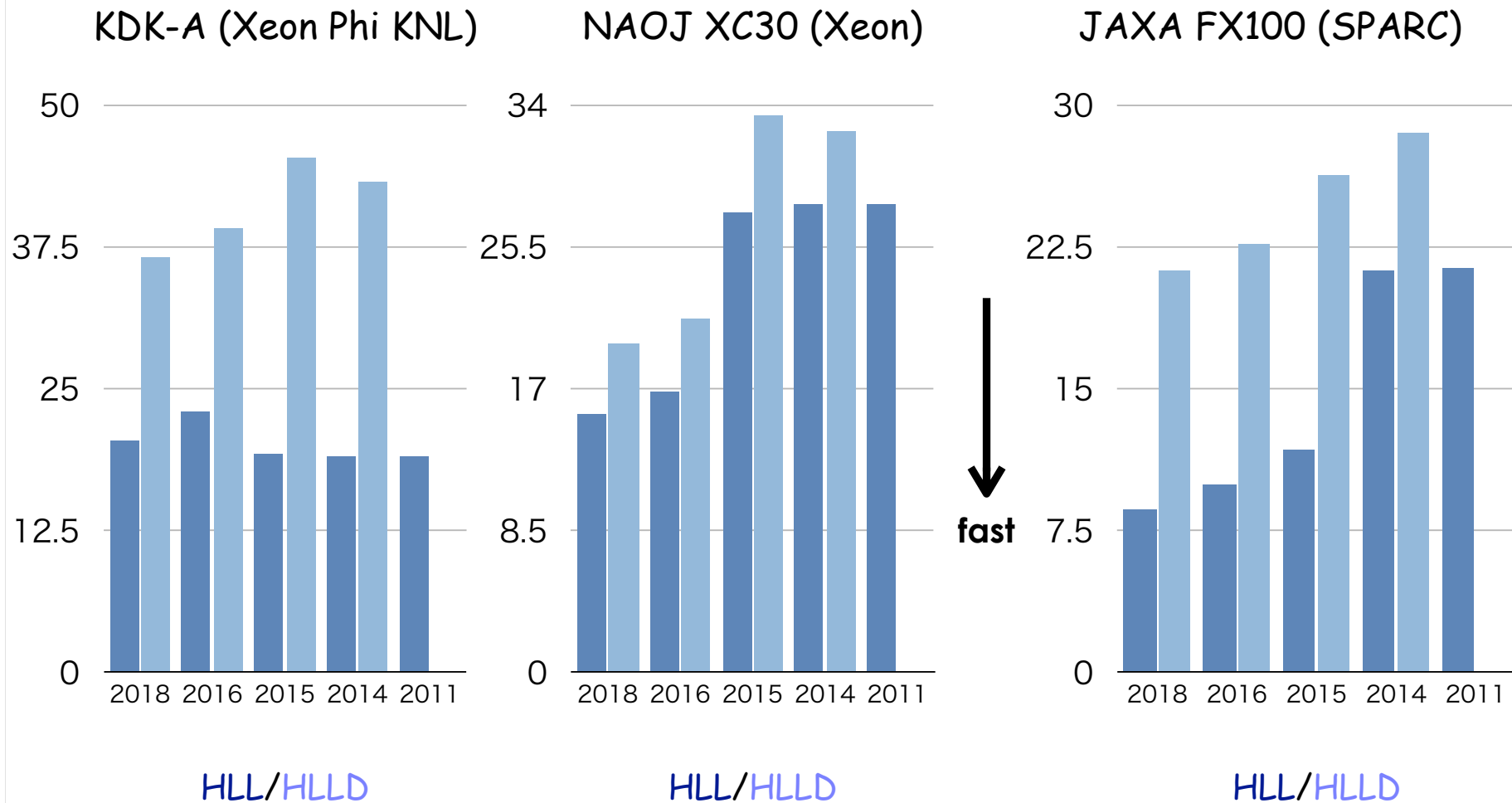


Parallel efficiency / strong scaling

- Very nice scaling up to 5000 cores; moderate scaling at 10000 cores
- 94% efficiency at 3000 cores with flat MPI
- 98% at 5000 physical cores with hyperthreading (@Xeon Phi KNL)



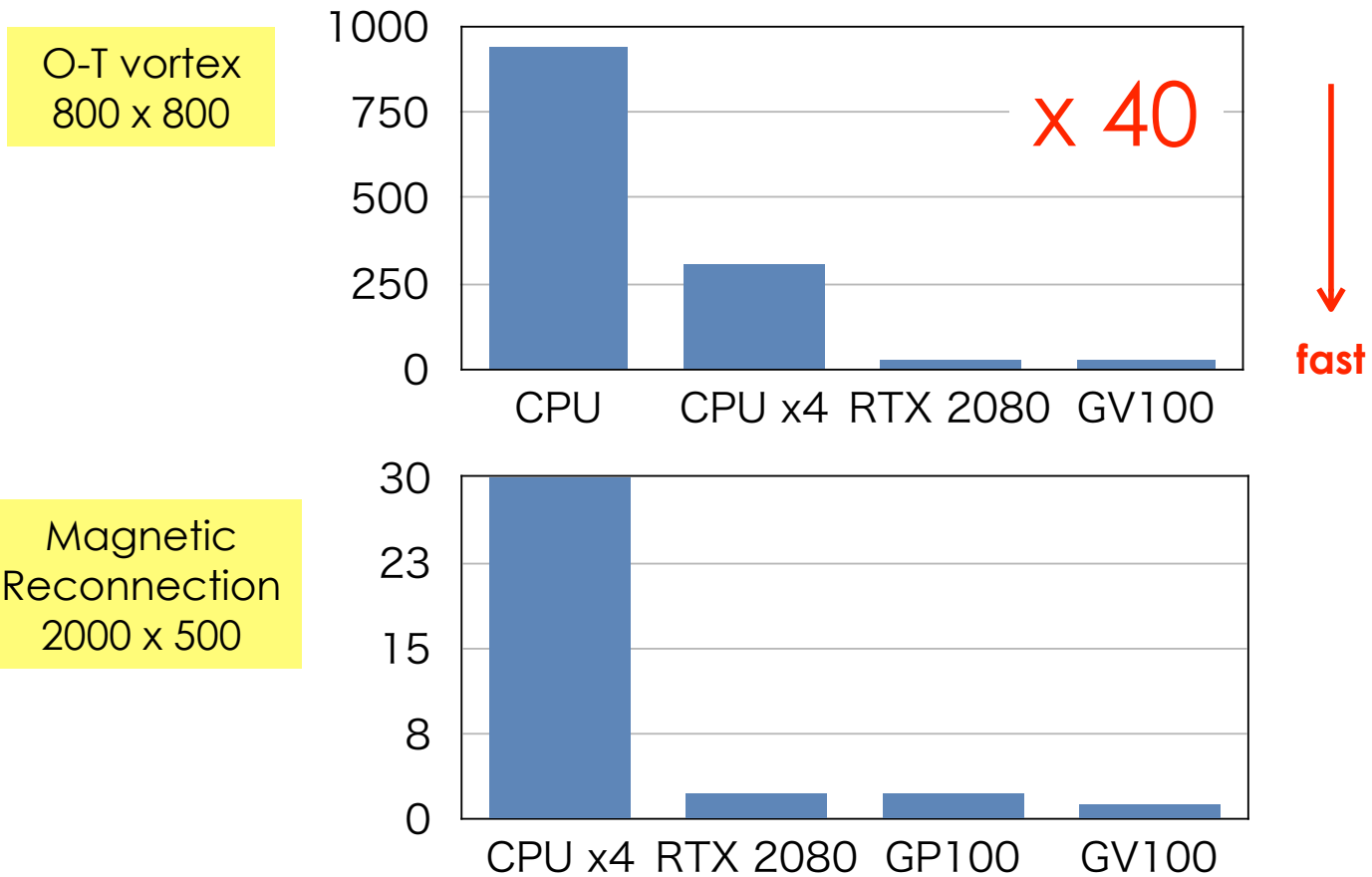
Performance: years of improvement



Optimized for various CPU architectures

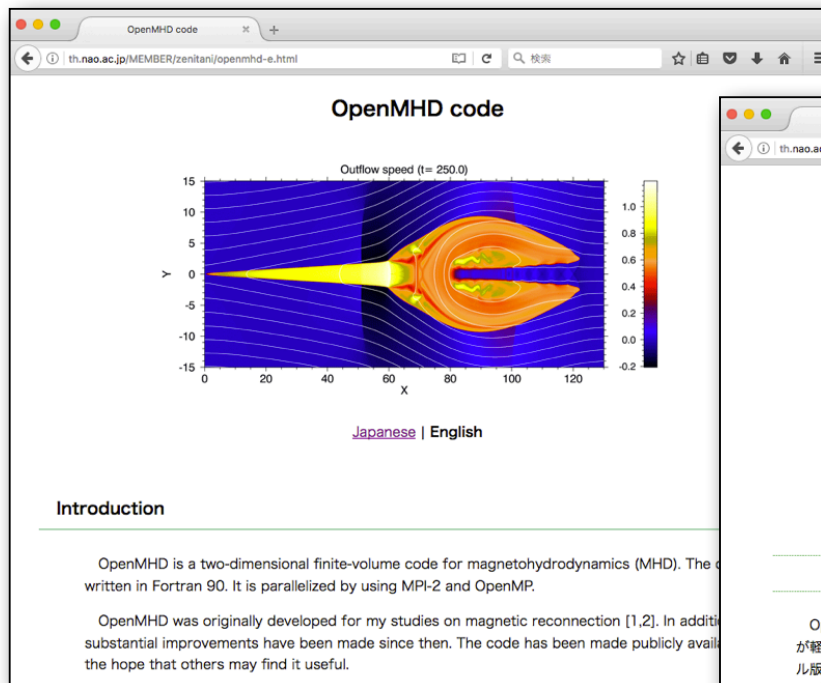
GPU version

- OpenMHD also runs on NVIDIA GPUs
- Partially written by CUDA Fortran



Web site

- Search “OpenMHD” with Google.
- Online documents [in Japanese]. English translation in progress.
- Gateway to a mailing list (41 users).



OpenMHD code

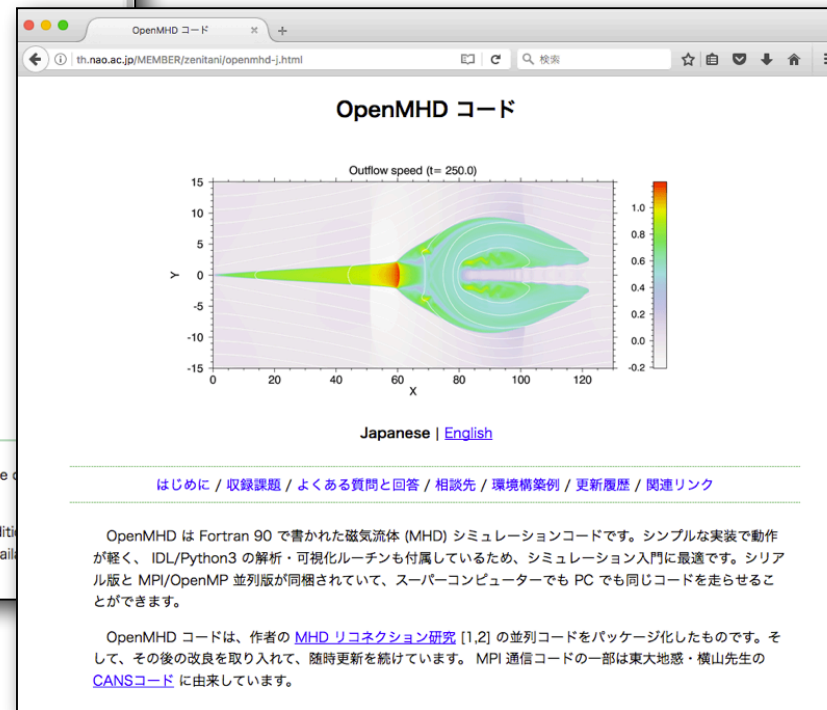
Outflow speed (t= 250.0)

Japanese | English

Introduction

OpenMHD is a two-dimensional finite-volume code for magnetohydrodynamics (MHD). The code is written in Fortran 90. It is parallelized by using MPI-2 and OpenMP.

OpenMHD was originally developed for my studies on magnetic reconnection [1,2]. In addition, substantial improvements have been made since then. The code has been made publicly available with the hope that others may find it useful.



OpenMHD コード

Outflow speed (t= 250.0)

Japanese | English

はじめに / 収録課題 / よくある質問と回答 / 相談先 / 環境構築例 / 更新履歴 / 関連リンク

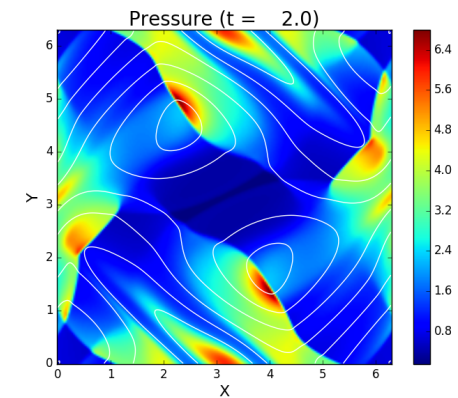
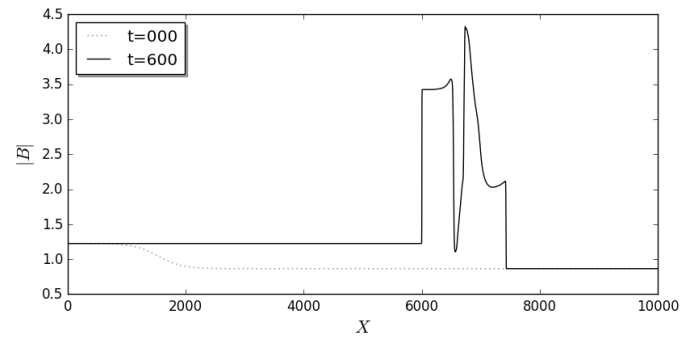
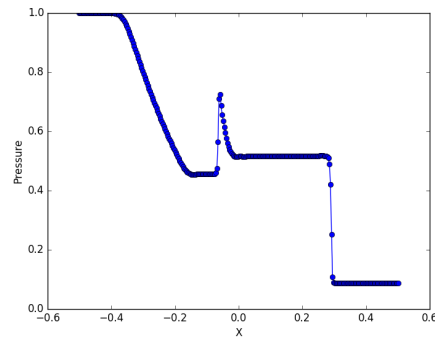
OpenMHD は Fortran 90 で書かれた磁気流体 (MHD) シミュレーションコードです。シンプルな実装で動作が軽く、IDL/Python3 の解析・可視化ルーチンも付属しているため、シミュレーション入門に最適です。シリアル版と MPI/OpenMP 並列版が同梱されていて、スーパーコンピュータでも PC でも同じコードを走らせることができます。

OpenMHD コードは、作者の [MHD リコネクション研究](#) [1,2] の並列コードをパッケージ化したものです。そして、その後の改良を取り入れて、随時更新を続けています。MPI 通信コードの一部は東大地惑・横山先生の [CANSコード](#) に由来しています。

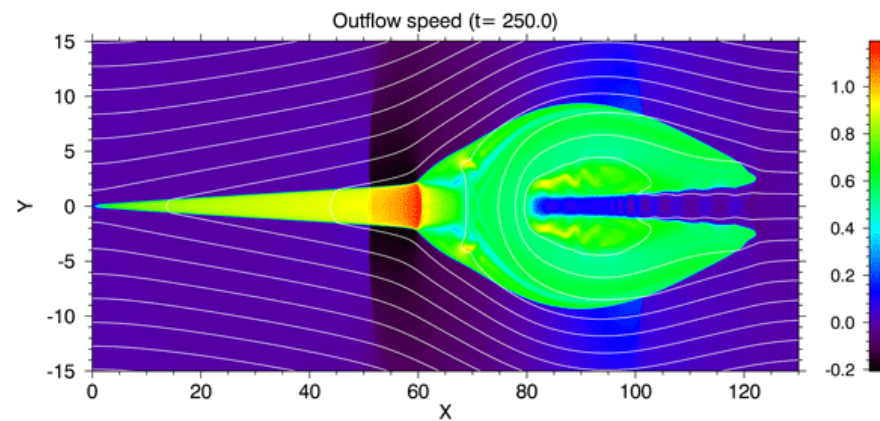
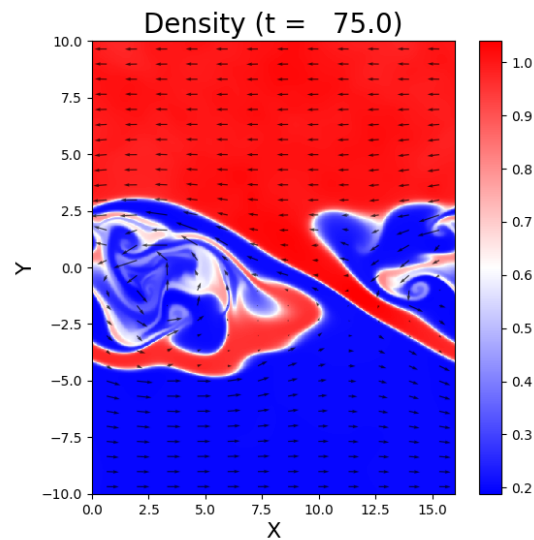
<https://sci.nao.ac.jp/MEMBER/zenitani/openmhd-e.html>

Built-in problems

- Riemann problem
- Solar-wind problem
- Orszag-Tang vortex



- KH & secondary instabilities
- Magnetic reconnection



Source code management

- Our repository is hosted by GitHub.
- It is open to public -- pull requests are always welcome!

The image shows two overlapping browser windows displaying the GitHub interface for the repository `zenitani/OpenMHD`.

The background window shows the repository overview page. It includes the GitHub navigation bar, repository name, and statistics: 82 commits, 1 branch, and 0 releases. A table lists the repository's subdirectories and their versions:

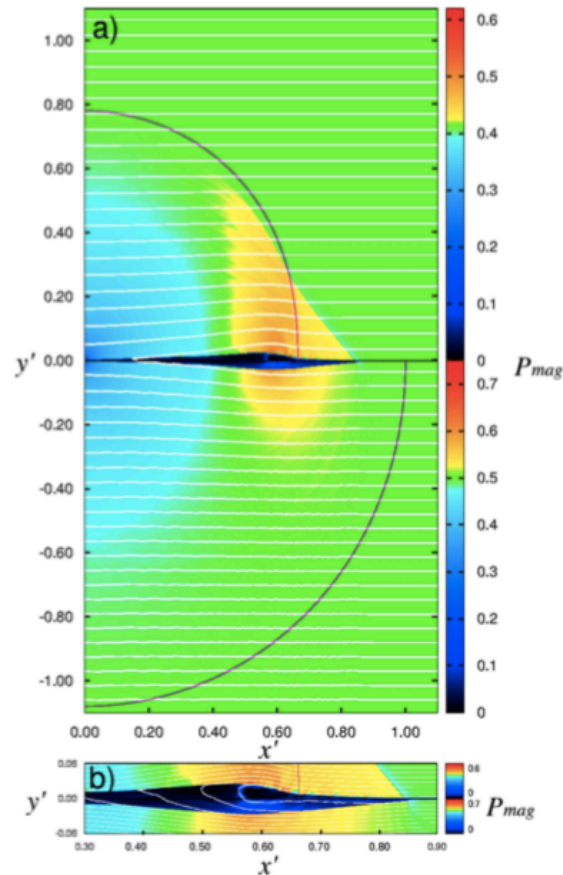
Directory	Version
1D_CIR	20170416
1D_basic	20170416
2D_KH	20170416
2D_basic	kind=mpi_offset_kind
2D_reconnection	20170416
visualization	20170416
COPYING	first commit
README.md	faster HLLC solver

The foreground window shows the file view for `OpenMHD / 2D_basic / flux_solver.f90`. It displays the file's commit history, contributor information, and the source code. The code is a Fortran subroutine for solving fluxes, with comments in Chinese and English. The code includes a header section with version information and a list of contributors with their dates and contributions:

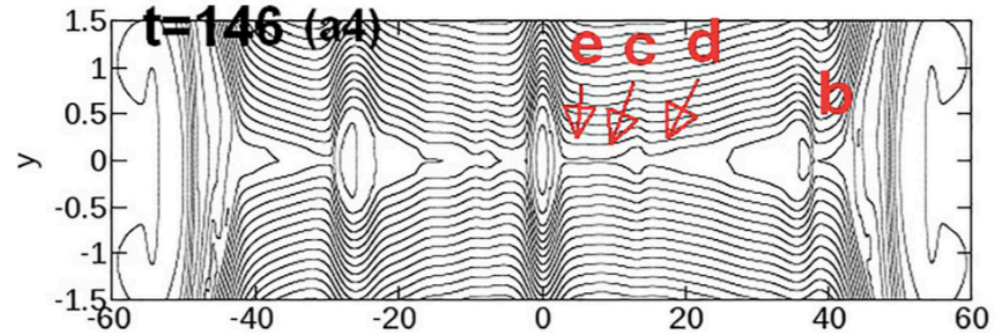
```
1 subroutine flux_solver(F,VL,VR,ix,jx,dir,type)
2 |-----|
3 | HLLC-G flux solver
4 | Ref: K. F. Gurski, SIAM J. Sci. Comput., 25, 2165 (2004)
5 | HLLD flux solver
6 | Ref: T. Miyoshi, K. Kusano, J. Comput. Phys., 208, 315 (2005)
7 |-----|
8 | 2010/05/11 S. Zenitani HLLC-G solver
9 | 2010/05/12 S. Zenitani HLLD solver
10 | 2010/09/18 S. Zenitani HLLD solver: fixed some bugs
11 | 2015/07/29 S. Zenitani HLL solver: if-statements ==> max/min functions
12 | 2015/08/15 S. Zenitani HLLC-G solver: optimization
13 | 2016/09/06 S. Zenitani X/Y/Z directions
14 |-----|
15 implicit none
16 include 'param.h'
17 |
18 | size of arrays [input]
19 integer, intent(in) :: ix, jx
20 numerical flux (F) [output]
```

Research papers based on OpenMHD

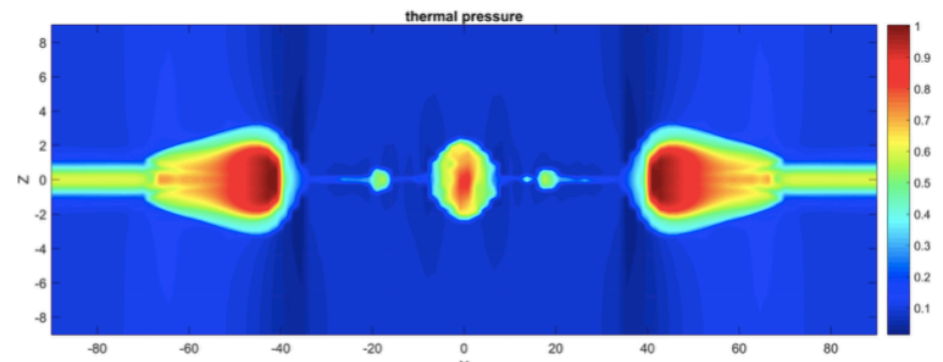
- Asymmetric reconnection
- Plasmoid-dominated reconnection



Nitta et al. 2016, 2019 ApJ



Shimizu et al. 2017 PoP



Hosseinpour et al. 2018 PoP

- Used in college & graduate school
- Kobe University and Ehime University

References

- [1] S. Zenitani & T. Miyoshi, *Phys. Plasmas* **18**, 022105 (2011)
- [2] S. Zenitani, *Phys. Plasmas* **22**, 032114 (2015)
- [3] A. Dedner et al., *J. Comput. Phys.* **175**, 645 (2002)
- [4] T. Miyoshi & K. Kusano, *J. Comput. Phys.* **208**, 315 (2005)
- [5] 錢谷誠司, 生存圏研究 13, 27 (2017)

URLs

- <https://ascl.net/1604.001>
- <https://github.com/zenitani/OpenMHD>
- <https://sci.nao.ac.jp/MEMBER/zenitani/openmhd-e.html>